CIRCUITS TO GENERATE A SEQUENTIAL INDEX FOR AN INPUT NUMBER IN A PRE-DEFINED LIST OF NUMBERS

Inventor: Madian Somasundaram

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to and incorporates by reference the entirety of the following U.S. provisional applications: United States Application Serial Number 60/445,150, entitled Memory And Power Efficient Longest-Prefix Match-Capable Content Addressable Memory, filed on March 15, 2003, United States Application Serial Number 60/442,779, entitled Efficient Implementation Of Content Addressable Memory Functions, filed on January 27, 2003, United States Application Serial Number 60/441,809, entitled Circuits To Generate A Sequential Index For An Input Number In A Pre-Defined List Of Numbers, filed on January 21, 2003.

FIELD OF INVENTION

[0002] The present invention relates to the field of memory devices and more particularly to the field of content addressable memories (CAMs).

BACKGROUND—DESCRIPTION OF PRIOR ART

[0003] The typical CAM is illustrated in Figure 1. An input number of width w (101) is compared against a table of allowed values (102). If there is a match, the location of the matched word (103) is returned. If the list of allowed values is 2^n long, the

CAM has to contain a minimum of 2^n x w bits of memory along with comparators, also 2^n x w in number.

[0004] It is also possible to perform the function of the CAM using a Random Access Memory, or RAM, using 2^w x n bits of RAM, as shown in Figure 2. The input number 201 is used to address a location in the memory 202 which contains the index 203 for that number, if it is in the list of allowed values. If the number is not in the list the corresponding index is value is set to some special value indicating no-match.

[0005] CAMs are expensive since each cell contains a memory element as well as a comparison circuit. Obtaining equivalent functionality with RAMs may require significantly more memory when the numbers are very wide.

[0006] The present invention improves upon these conventional CAMs. The system and method disclosed here can compare inputs of width w against a list of length up to 2^w using only about 2^w bits of RAM. As a result, considerable cost savings are possible compared to previous implementations. Number ranges and arbitrary groupings of numbers are possible using extensions of the methods of the disclosure. The application of this invention extends beyond searches and lookups. For example, the method could be used to compress symbols in a communication stream.

SUMMARY OF THE INVENTION

[0007] Methods are disclosed to determine whether a given input word is found in a list of numbers, or number ranges, and if found, the present invention generates an index that is in a compact range equal exactly to the number of elements in the list.

DRAWINGS

[0008] FIG.1 is a schematic of a CAM used to look up a number in a list of numbers.

[0009] FIG.2 shows how a RAM can be used to obtain the function of a CAM.

[0010] FIG.3 is a schematic of the method according to one embodiment of the present invention.

[0011] FIG.4 is an example table of numbers.

[0012] FIG.5 shows the value of bits used for implementing the table of Figure 3.

[0013] FIG.6 shows how two example input numbers are processed according to one embodiment of the present invention.

[0014] FIG.7 is an example table with a set of numbers in each element of the table.

[0015] FIG.8 shows the value of bits used for implementing the table of Figure 6.

[0016] FIG.9 shows how an example input number is processed according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0017] Figure 3 is an embodiment of the invention to lookup inputs against of table of 2^w elements. 2^w bits of memory are arranged in rows of length 2^b bits (301).

There are 2^{w-b} rows. Each row also has an associated value, called the "Last Index", of length w+1 bits (302.) The w-b high order bits (303) of the input word (304) are used to select a particular row in the memory, and that word is output on lines 305, and the Last Index is output on lines 306. The b low order bits of the input word (307) are used to indicate a particular bit position in word read out. If the indicated bit is not set, the circuit generates an invalid signal (308.) If the bit is set, a population count circuit (309) counts all the ones in the word up to the selected bit and the result appears in lines 310. The adder 311 adds the population count to the Last Index of the selected row. The result 312 is the index of the input word in the list.

[0018] The embodiment described requires only approximately 2^w RAM elements for lists of length up to 2^w and width w. An equivalent CAM would require up to (2^w x w) CAM elements. Not only are fewer elements used (smaller by a factor of w), the RAM elements used are much cheaper than the CAM elements, resulting in a tremendous overall cost savings. Compared to prior art RAM implementations, the method shown here reduces the number of elements, and hence the cost by a factor w.

Operation

[0019] The operation is illustrated by considering table in Figure 4 which shows a list of 8 valid numbers, each 8 bits long. The valid values are 7, 8, 73, 102, 119, 128, 129, and 193. These values loaded into an embodiment of the invention with 16 rows, each with 16 bits, and a Last Index value of 4 bits as shown in the table of Figure 5. All bits of the table of Figure 5 are 0 except for the 7^{th}, 8^{th}, 73^{rd}, 102^{nd}, 119^{th}, 128^{th}, 129^{th},

and 193rd bits which are set to 1. The Last Index is set to the total of the number of bits set in all previous rows reduced by 1.

[0020] The table in Figure 6 shows two illustrative input words, and the steps to arrive at the index. The first column shows the processing of an input word that is found in the list, i.e. 0100 1001. When the input word is presented to the circuit, the 4 high order bits (in this instance 0100) are used to address the rows, and thus row 4 is selected. The low order bits (in this instance 1001) are used to select a position in the row, and thus position 9 is selected. Since the bit in the selected position is 1, the input word is in the table. The bits in the row up to position 9, i.e. bits 0 through 9 are added up by the population count circuit. The sum is added to the Row Subtotal for that row, and the result gives the sequential index of the input number. The second column shows the processing an input word that is not found in the list, i.e. 1100 0011. When the input word is presented to the circuit, the 4 high order bits (in this instance 1100) are used to address the rows, and thus row 12 is selected. The low order bits (in this instance 0011) are used to select a position in the row, and thus position 3 is selected. Since the bit in the selected position is 0, it indicates that the input number is not part of the table, and hence an invalid signal will be generated.

[0021] The invention can be extended by adding additional bits to the table and by changing the processing of a selected row. An example extension allows mapping of a set of numbers in a given range of numbers. The set of numbers need not be contiguous as long as no other index in the same list is mapped to a number in the range.

[0022] Sets of numbers are handled by adding an additional bit to each position in a row. The first bit indicates if the position is a valid number in the list, while

the second bit indicates whether it is the start of the set of numbers. The population count circuit only counts the number of start-of-set bits up to the selected position, and the Last Index has the count of all start of range bits before the selected row reduced by 1. The table of Figure 7 shows an example with number ranges. The assignment of bits in an embodiment of the invention is shown in Figure 8. The first bit in each location is set if the corresponding number is in the list. Thus considering the values for index value 1, the locations 11 through 20, 22, and 25 through 35 are set to 1. The second bit of each location is set only for the first location of each index. Thus for index 1, the second bit is set only in location 11. Figure 9 shows the processing of an example input value, 1101 0000. The row chosen is 13, and the bit address is 0. The 1st bit is set at this location, so the input is in the list. In order to compute the index, the 2nd bit is added up to the selected location, resulting in an index of 5.